Corso di Architettura dei Sistemi a Microprocessore

Memory Systems: Cache e Memoria Virtuale

Luigi Coppolino

Contact info

Prof. Luigi Coppolino
luigi.coppolino@uniparthenope.it

Università degli Studi di Napoli "Parthenope"
Dipartimento di Ingegneria

Centro Direzionale di Napoli, Isola C4
V Piano lato SUD - Stanza n. 512

Tel: +39-081-5476702
Fax: +39-081-5476777

References

- Textbook: chapter 8

Argomenti della Lezione

- DMA (Direct Memory Access)
- Gerarchie di Memorie, concetti e dettagli
- Memorie Cache
- Memoria Virtuale

Basic Concepts

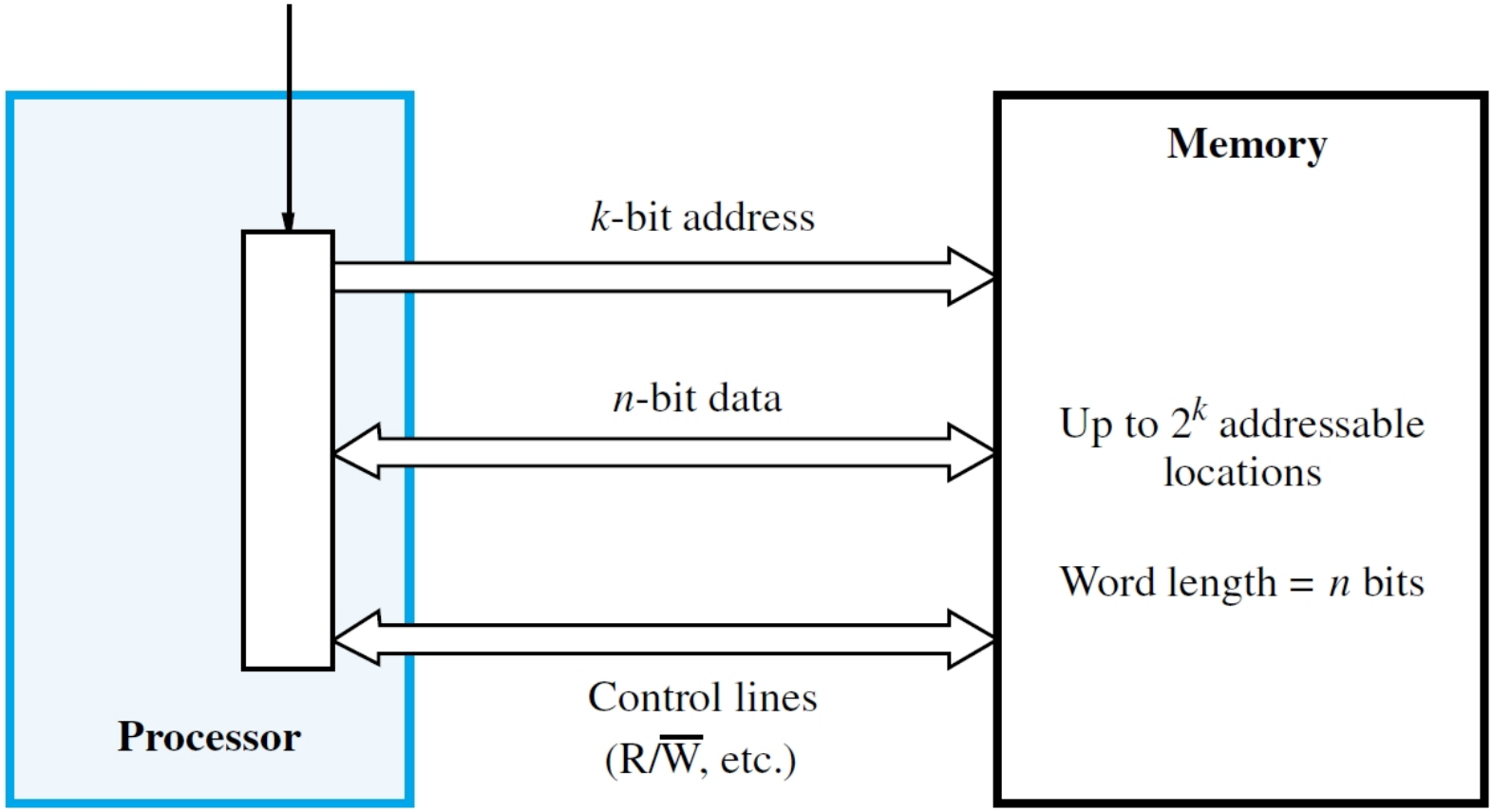
- Abbiamo introdotto il concetto di “memoria principale” o memoria RAM
  - Access Time indipendente dalla posizione del dato in memoria

Memory access time è il tempo che intercorre tra l’inizio e il completamento del trasferimento di una Word/Byte

Memory cycle time è l’intervallo minimo che intercorre tra due accessi successivi alla memoria

- Abbiamo inoltre introdotto l’organizzazione indirizzo/dato/controllo dell’interfaccia tra processore-memoria

Interfaccia Memoria-Processore



Cache and Virtual Memory

- The main memory is slower than processor
- Cache memory is smaller and faster memory that is used to reduce effective access time
- Holds subset of program instructions and data
- Information for one or more active programs may exceed physical capacity of the memory
- Virtual memory provides larger apparent size by transparently using secondary storage
- Both approaches need efficient *block transfers*

Direct Memory Access (DMA)

- Program-controlled I/O richiede un continuo coinvolgimento del processore
- Con ogni interazione viene trasferito una sola word (o byte) con un enorme oberhead
- I traferimenti tipicamente coinvolgono grossi blocchi di ati

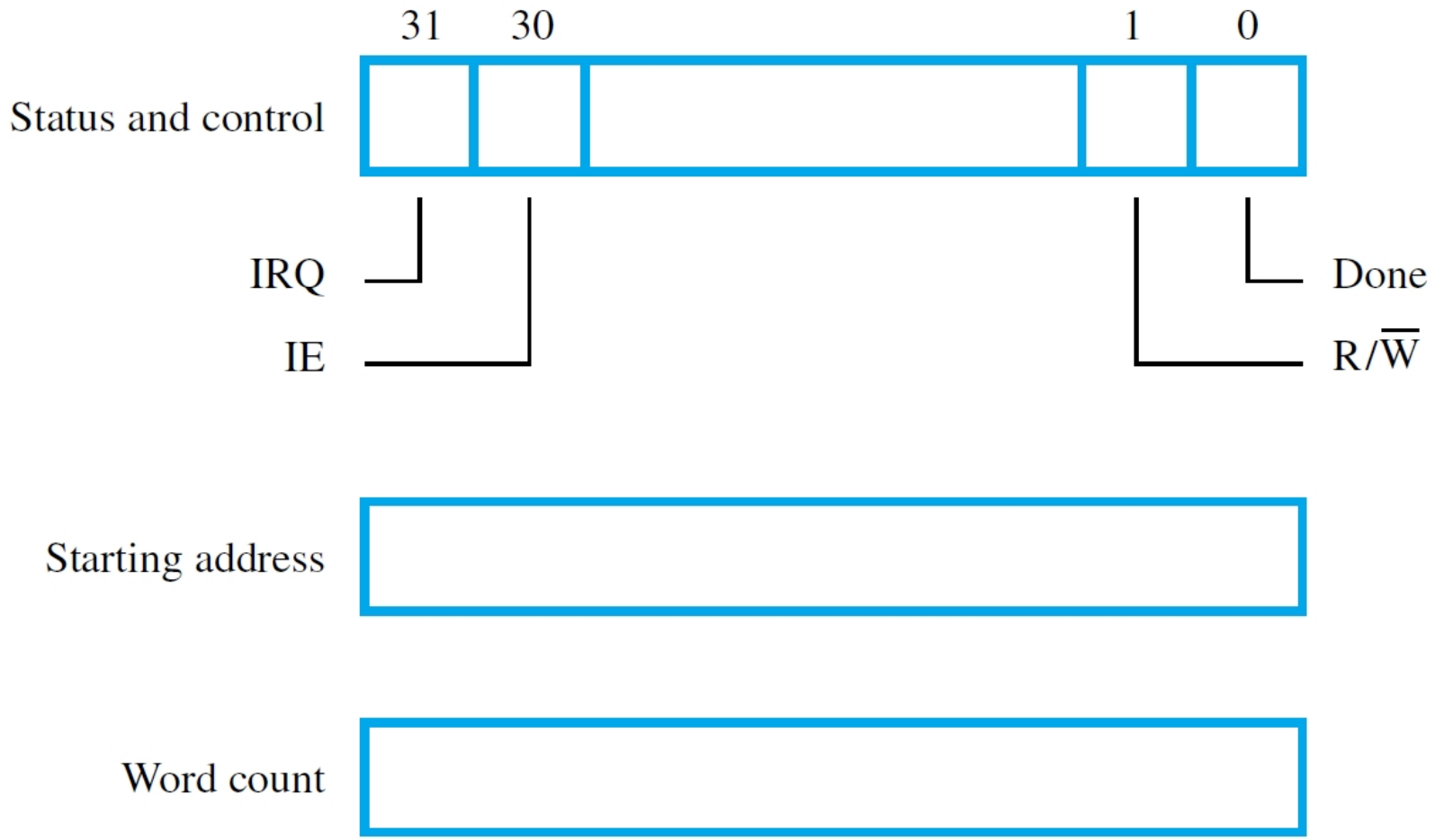
- Alternativa: direct memory access (DMA)

Un device dedicato al trasferimento di *blocchi* di dati tra memoria e altri dispositivi di I/O

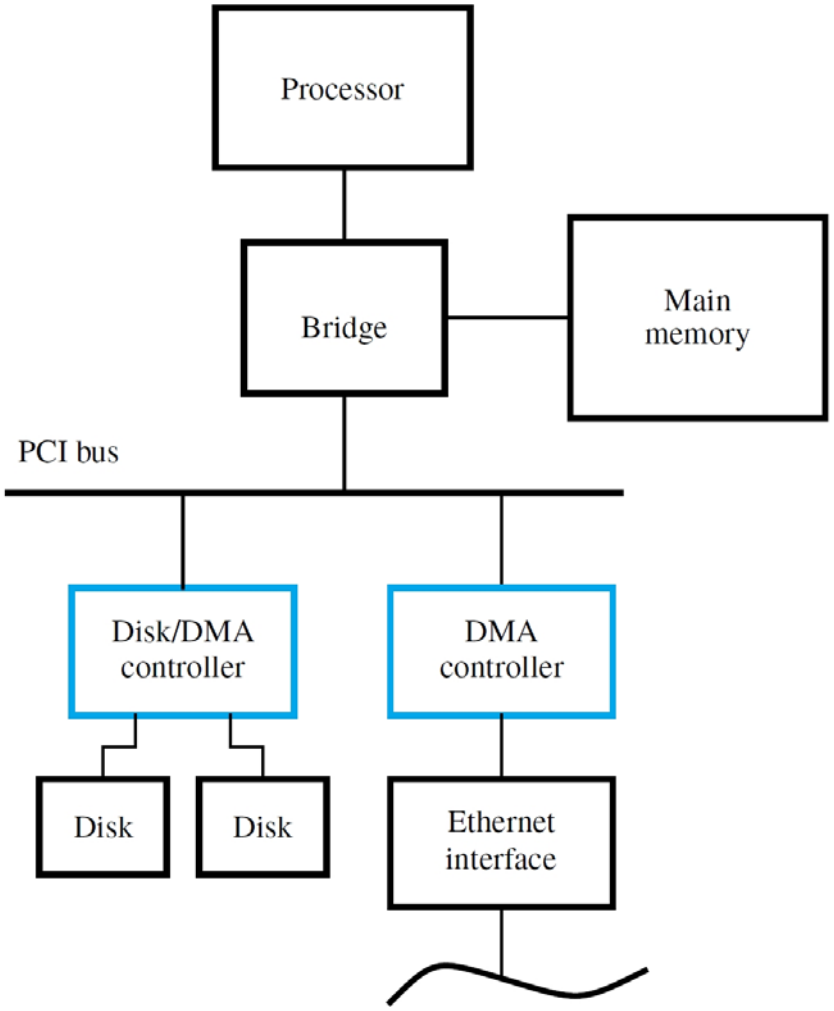
DMA Controller

- *DMA controller* può essere condiviso o specifico per un particolare I/O device
- Effettua i singoli trasferimenti che avrebbe gestito il processore tendendo traccia del numero di byte/word trasferiti
- Il processore inizializza le operazioni di trasferimento attraverso registri del DMA
- Quando termina il suo task, il DMA avverte il Processore generando un'interruzione
- Esempi di dispositivi che usano DMA controller: disk e Ethernet

Interfaccia DMA device



Esempio di configurazione con due DMA



Gerarchie di Memoria

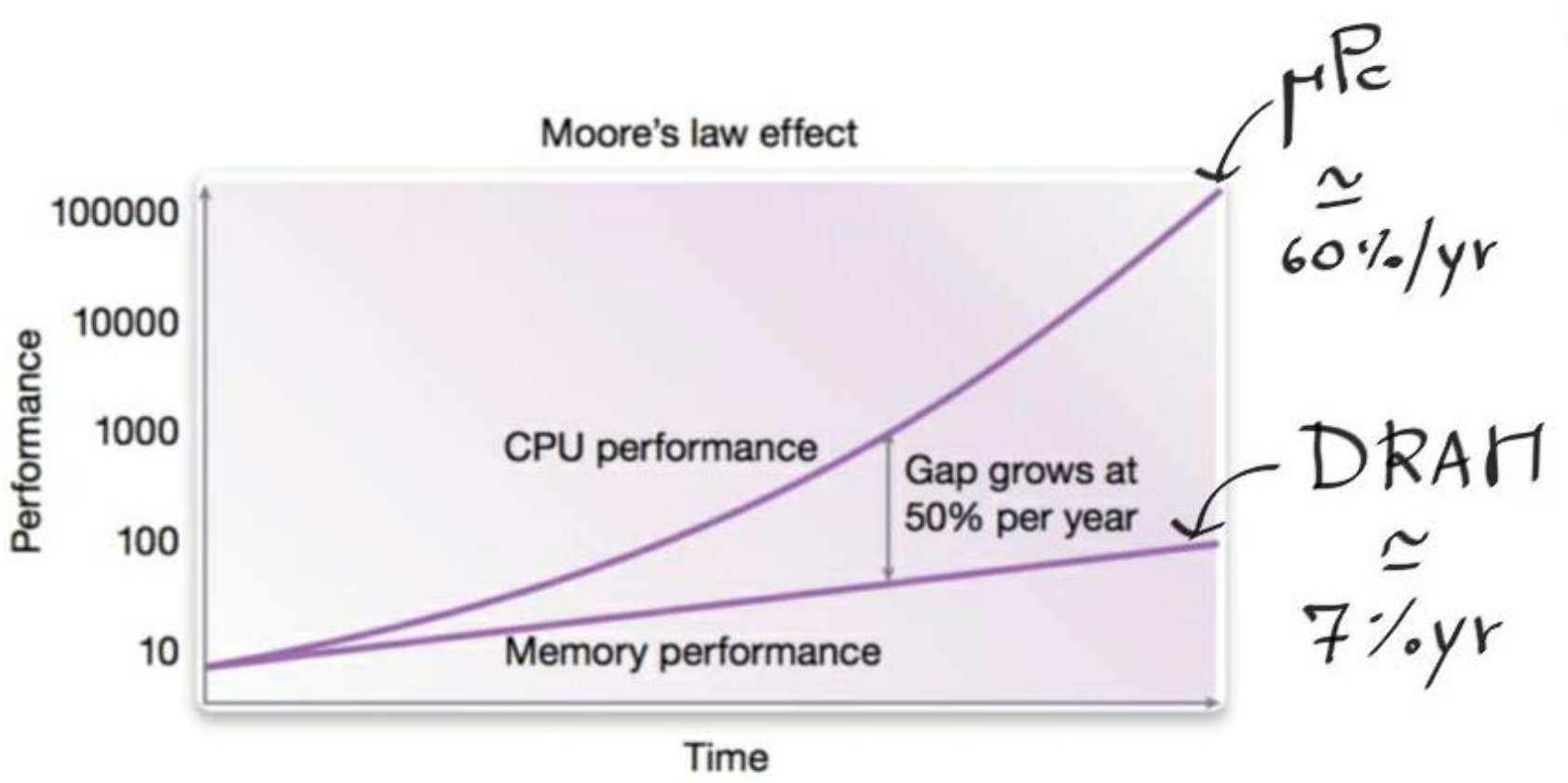
- La memoria ideale è veloce, grande ed economica
- Una tale memoria è non può esistere

Utilizzare una gerarchia di memorie

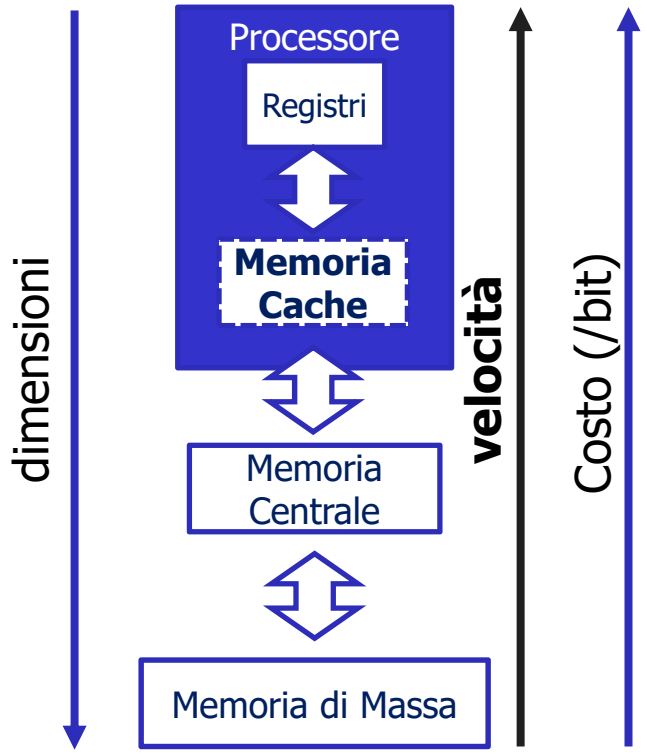
- Usare le caratteristiche del software per far apparire la memoria "grande" e "veloce"

- Memorie piccole e veloci vicine al processore (statiche)
- Memorie più grandi e lente (dinamiche)
- Memorie più lente per mass storage

Rationale


Moore's law effect
Performance
100000
10000
1000
100
10
CPU performance
Gap grows at
50% per year
Memory performance
Time
μPc
≃
60%/yr
DRAM
≃
7%/yr

Memoria Cache



- Ha dimensioni ridotte (KB-MB)
- È veloce (SRAM)
- È invisibile al processore
  - Che continua a generare indirizzi in maniera trasparente
- Non può ospitare l'intero contenuto della RAM per cui sarà necessario
  - Definire quale porzione della RAM caricare
  - Come sostituire il contenuto della cache

Il tutto funziona se la maggior parte degli accessi alla memoria trova effettivamente i dati in cache

Principi di località

Dalle evidenze sperimentali:

PRINCIPIO DI LOCALITA' SPAZIALE: se al tempo T si verifica un accesso all'indirizzo A, è altamente probabile che l'indirizzo acceduto al tempo T+1 sia in un intorno di A

PRINCIPIO DI LOCALITA' TEMPORALE: se al tempo T si verifica un accesso all'indirizzo A, è altamente probabile che l'indirizzo A sia nuovamente acceduto nell'immediato futuro di T

Le evidenze suggeriscono che:
1) conviene lavorare su **blocchi di dati** anziché singole parole
2) conviene tenere in cache i dati usati di recente

Interazioni Processore-Cache

- Il processore, durante una **lettura** o una **scrittura**, genererà un indirizzo come farebbe in assenza di cache
- Il dato corrispondente all'indirizzo può essere presente in cache oppure no
  - Cache hit
  - Cache miss

CACHE HIT: Il dato richiesto è in cache

- **Caso LETTURA**:
  - Il dato viene prelevato dalla cache e inviato al processore: nessuna interazione con la memoria principale
- **Caso SCRITTURA**:
  - Scrittura in cache => perdita di consistenza
  - Due possibili policy (Write policy):
  1. **Write-Through**: la scrittura viene effettuata sulla cache e contemporaneamente sulla memoria principale
     - Semplice, ma il tempo di scrittura è dettato dalla velocità della memoria principale
  2. **Write-Back**: la scrittura avviene in cache, ma il blocco modificato è invalidato (valid bit), quando il blocco sarà cacciato dalla cache, se non valido, sarà ricopiato sulla memoria principale
     - Necessario aggiungere un bit per ciascun blocco della cache, con il vantaggio che più scritture sullo stesso blocco generano un solo accesso alla memoria (quando sarà scaricato il blocco).

CACHE MISS: IL DATO RICHIESTO NON E' PRESENTE IN CACHE

- CASO LETTURA: dato trasferito in cache
- CASO SCRITTURA:
  - **Write-Through**: dato modificato nella Memoria Principale (eventualmente portato in cache)
  - **Write-Back**: dato trasferito in cache e poi modificato

DATO TRASFERITO IN CACHE:

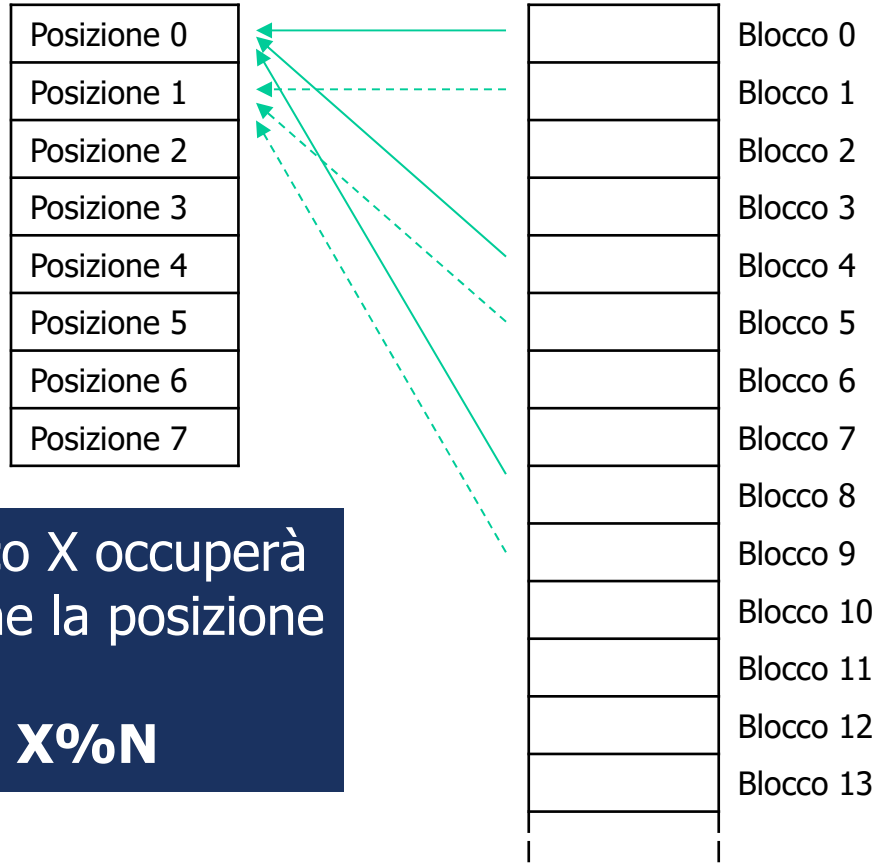
- Dove? => Politica di Mapping
- Se la cache è piena, quale blocco dovrà essere liberato per far posto al blocco entrante? => Politica di Sostituzione

POLITICA DI MAPPING

- Definisce la corrispondenza tra indirizzi nella memoria principale e indirizzi nella cache
- Tre politiche:
  - Mapping Diretto (Direct Mapping)
  - Mapping completamente Associativo (Full Associative)
  - Mapping Set-Associativo

Direct MAPPING

Cache di N=8 blocchi



Il blocco X occuperà in Cache la posizione

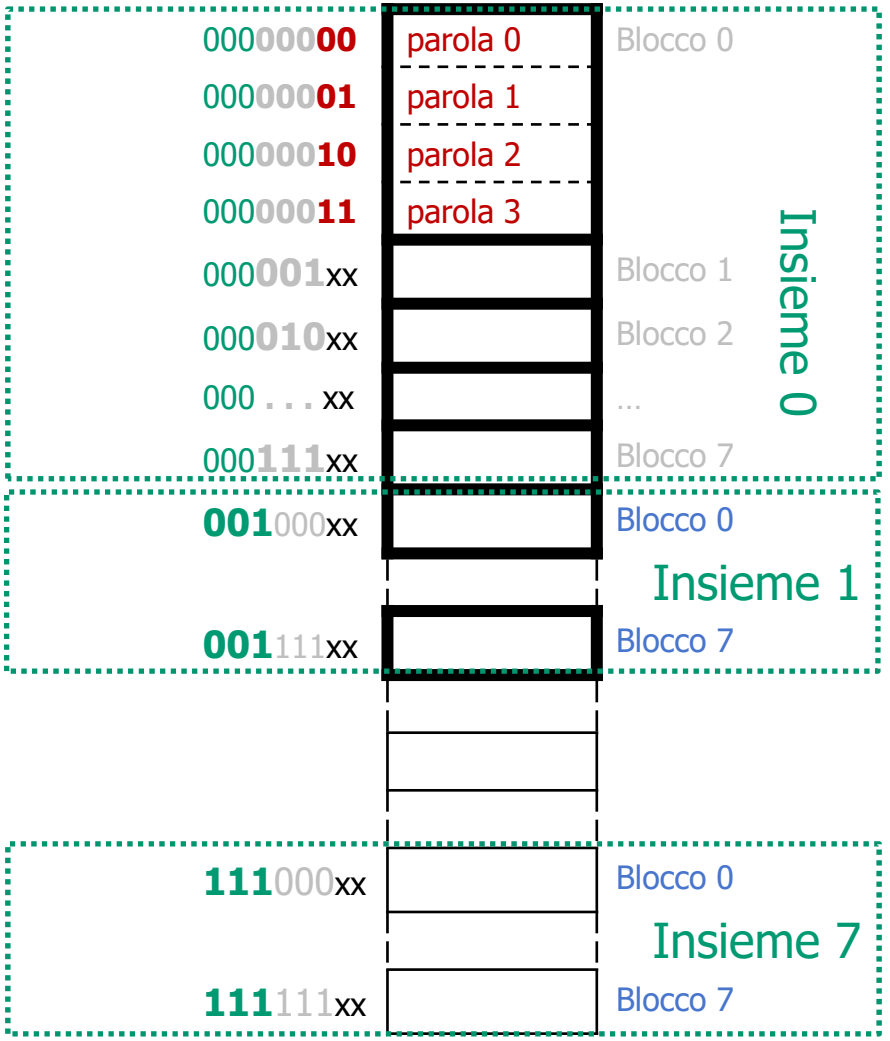
POS = X%N

Direct MAPPING: ACCESSO AL DATO
memoria da 256 (2^8) parole
CACHE DA 8 (2^3) Blocchi da 4 (2^2) parole ciascuno

ETICHETTA (3 bit)	BLOCCO (3 bit)	SPIAZZAMENTO (2 bit)
101	110	11

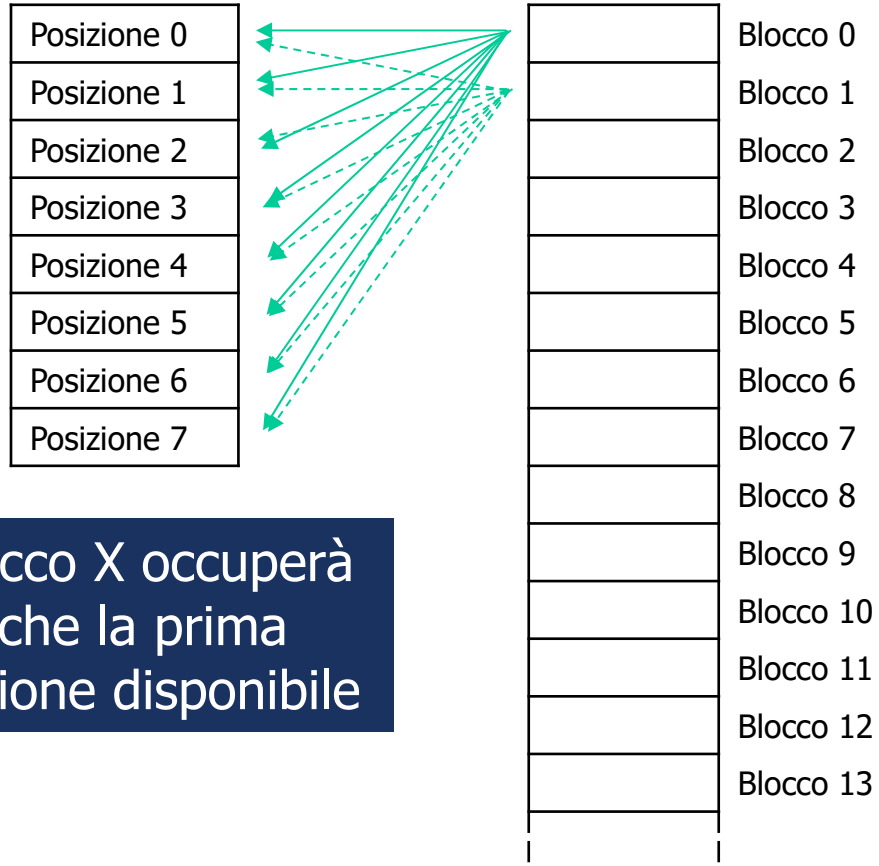
ETICHETTA	Posizione 0
ETICHETTA	Posizione 1
ETICHETTA	Posizione 2
ETICHETTA	Posizione 3
ETICHETTA	Posizione 4
ETICHETTA	Posizione 5
ETICHETTA	Posizione 6
ETICHETTA	Posizione 7 (N-1)

Cache di N=8 blocchi da 4 parole



FULL ASSOCIATIVE

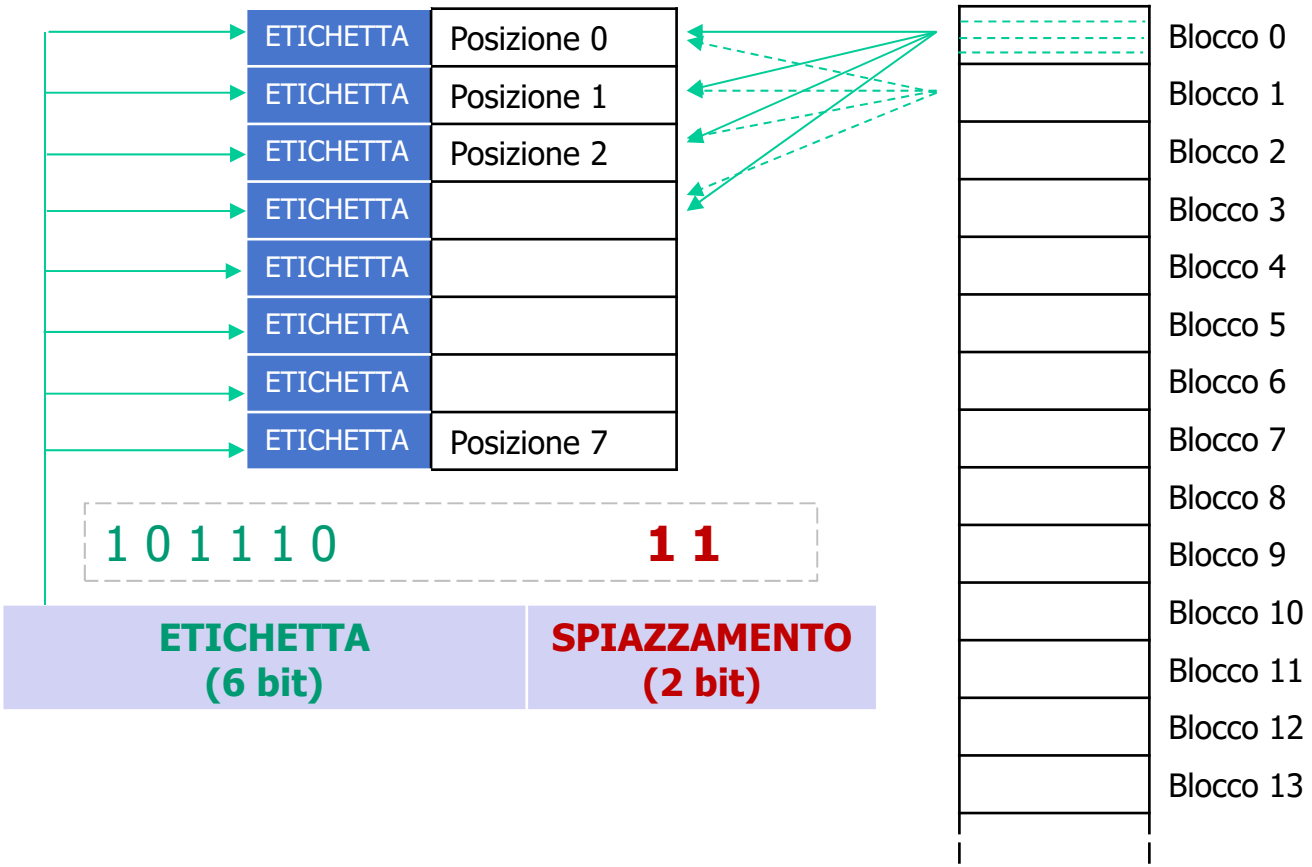
Cache di N=8 blocchi



Il blocco X occuperà in Cache la prima posizione disponibile

FULL ASSOCIATIVE: ANALISI DELL'INDIRIZZO

Cache di N=8 blocchi da 4 parole

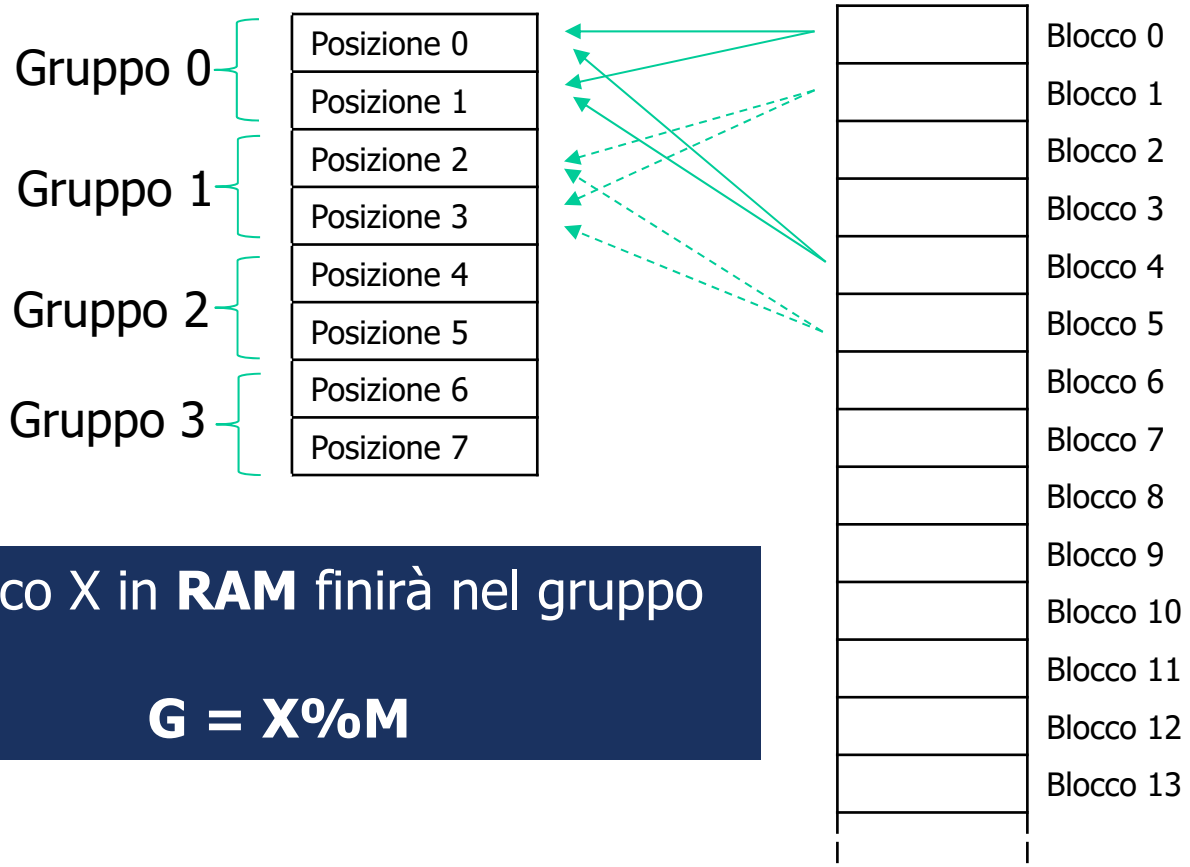


Associative Mapping: Considerazioni

- Massima Flessimbilità
- L'Etichetta usa tutti i bit non necessari ad identificare la parola nel blocco
- Il confronto dell'etichetta della parola cercata è effettuato in parallelo su tutti i campi delle entry della cache
- Maggior complessità dell'HW
- Quando si riempie la cache è necessario un algoritmo per identificare il campo da liberare

SET ASSOCIATIVA

Cache di N = 8 blocchi in M = 4 gruppi



Il blocco X in **RAM** finirà nel gruppo

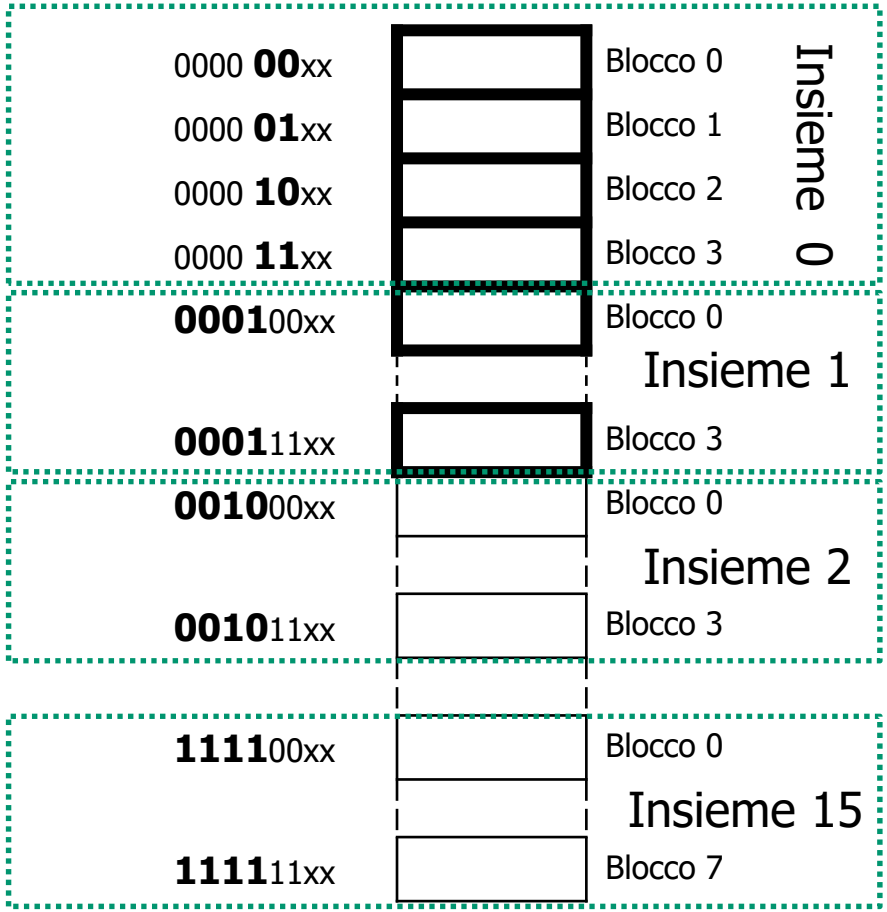
G = X%M

SET ASSOCIATIVA: ACCESSO AL DATO

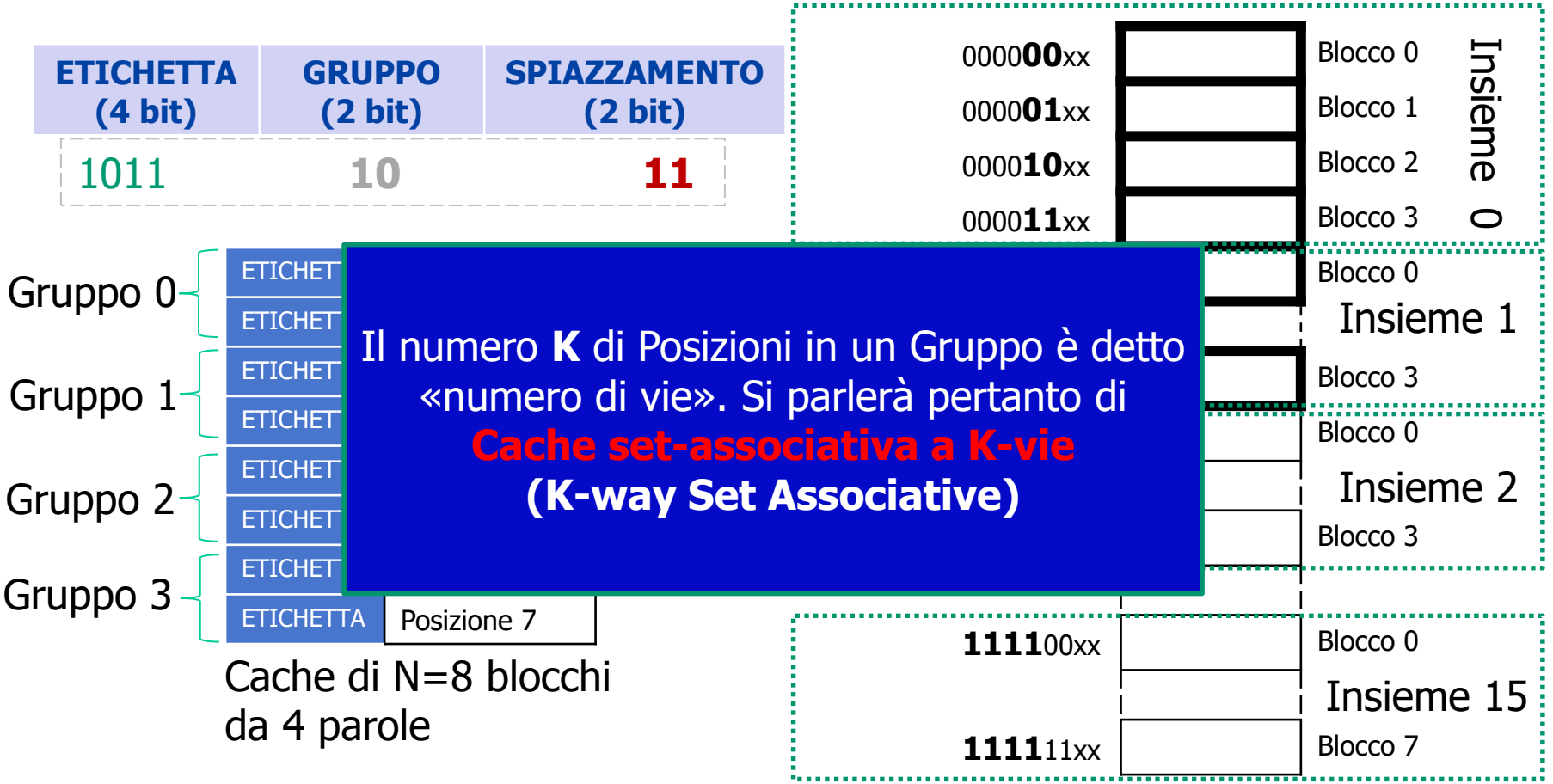
ETICHETTA (4 bit)	GRUPPO (2 bit)	SPIAZZAMENTO (2 bit)
1011	10	11

Gruppo 0	ETICHETTA	Posizione 0
	ETICHETTA	Posizione 1
Gruppo 1	ETICHETTA	Posizione 2
	ETICHETTA	Posizione 3
Gruppo 2	ETICHETTA	Posizione 4
	ETICHETTA	Posizione 5
Gruppo 3	ETICHETTA	Posizione 6
	ETICHETTA	Posizione 7

Cache di N=8 blocchi da 4 parole



SET ASSOCIATIVA: ACCESSO AL DATO



Set-Associative Mapping: Considerazioni

- Combina direct & associative mapping
- Raggruppa i blocchi della cache in *sets (gruppi)*
- Il campo Blocco ora mappa il blocco su un unico gruppo
  - Ogni blocco nel
- Associative search involves only tags in a set
- Replacement algorithm is only for blocks in set
- Reducing flexibility also reduces complexity
- *k* blocks/set $\rightarrow$ *k*-way set-associative cache
- Direct-mapped = 1-way; associative = all-way

Politica di SOSTITUZIONE: quale blocco liberare?

- Nel caso Direct Mapping la soluzione è banale
- Nel caso di cache associative:
  - Idealmente: sostituire il blocco che **non sarà usato** nel prossimo futuro
    - Corollario Località Temporale: Il dato usato meno recentemente è quello che meno probabilmente sarà usato nel futuro
    - Policy LRU (Least Recently Used): ad ogni blocco associato un contatore azzerato ad ogni accesso al blocco
  - LRU efficace ma costoso
  - Random: si mostra essere molto efficace ed è sicuramente poco costoso

Performance ANALYSIS

- **HIT RATE** (**h**): rapporto tra gli hit in cache e il totale degli accessi in memoria (ipotizziamo 95%)
- **MISS RATE (1-h):** rapporto tra i miss in cache e il totale degli accessi in memoria (5%)
- **C** = tempo di risposta della cache
- **R** = tempo di accesso alla RAM (inclusa una penalty) (ipotizziamo 10*C)
- **N** = Numero di accessi in memoria
- $\mathbf{T_{NO\_CACHE} \approx N*R}$ (tempo per l'accesso ai dati in assenza di cache)
- $\mathbf{T_{CACHE} = N*h*C+N*(1-h)*R}$ (tempo per l'accesso ai dati in presenza di cache)

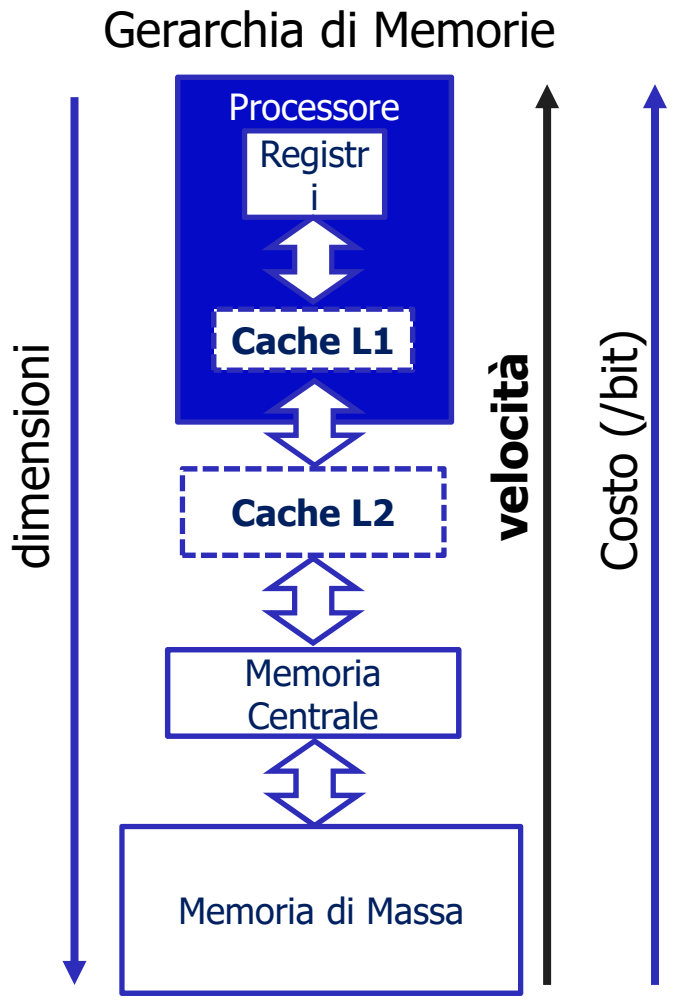
- $\text{SPEED\_UP} = \frac{T_{NO\_CACHE}}{T_{CACHE}} = \frac{N*R}{N*h*C+N*(1-h)*R} = \frac{10*C}{0,95*C+0,05*10*C} = \frac{10}{0,95+0,05*10} =$ 6,90

Le Cache nei Sistemi Moderni

Livelli Multipli di Cache

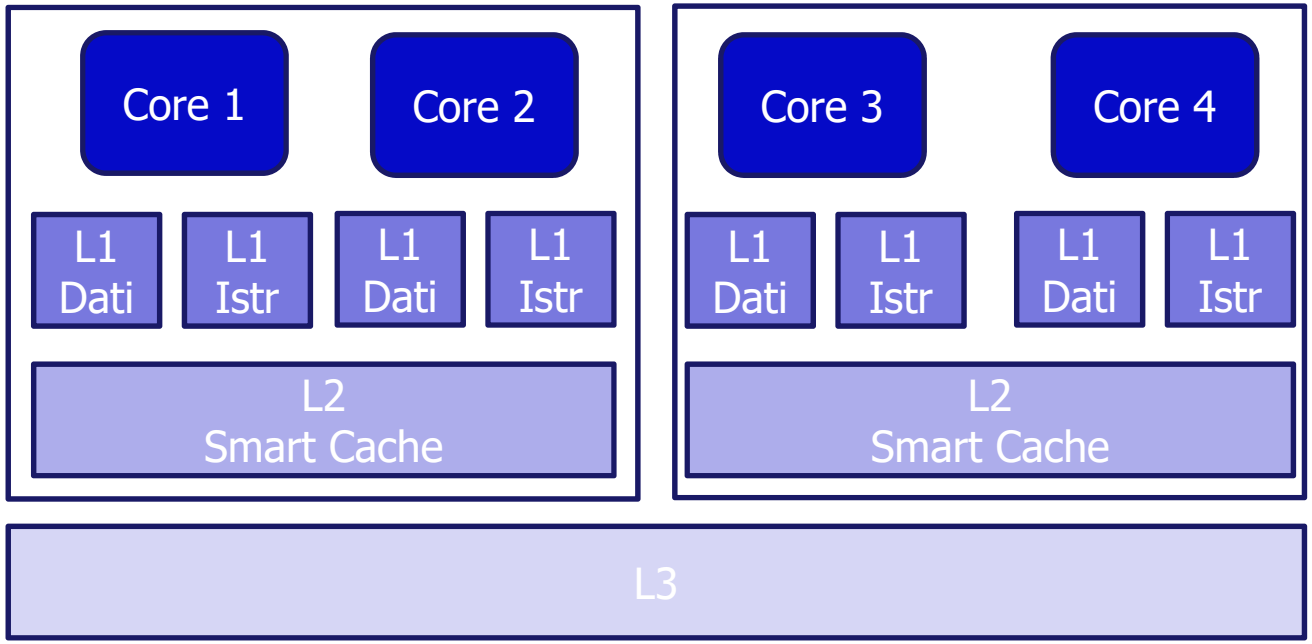
- la L1 serve gli HIT, la L2 serve i MISS.

MISS a due livelli = MISS RATE L1*MISS RATE L2



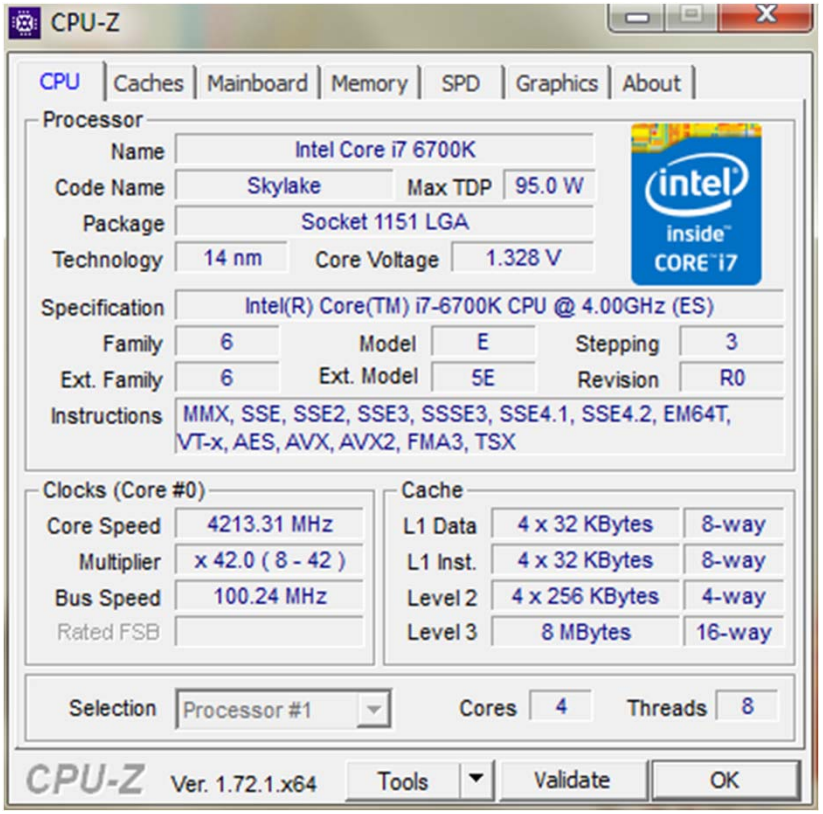
Intel Cache Levels

- Intel introduce Cache con 486Dx
- Dal Pentium II introdotta L2 Cache
  - Con il Pentium III la L2 passa sul Chip del processore
- Con il Pentium IV introdotta la L3
- Intel I7:



Intel i7-6700 (Skylake), 4.0 GHz (Turbo Boost), 14 nm

- Configurazione
  - L1 Data cache = 32 KB, 8-WAY.
  - L1 Instruction cache = 32 KB, 8-WAY.
  - L2 cache = 256 KB, 4-WAY
  - L3 cache = 8 MB,16-WAY
- Latenza
  - L1 Data Cache Latency = 4/5 cycles
  - L2 Cache Latency = 12 cycles
  - L3 Cache Latency = 38 cycles
  - RAM Latency = 42 cycles



Esercizio 1 (Svolto)

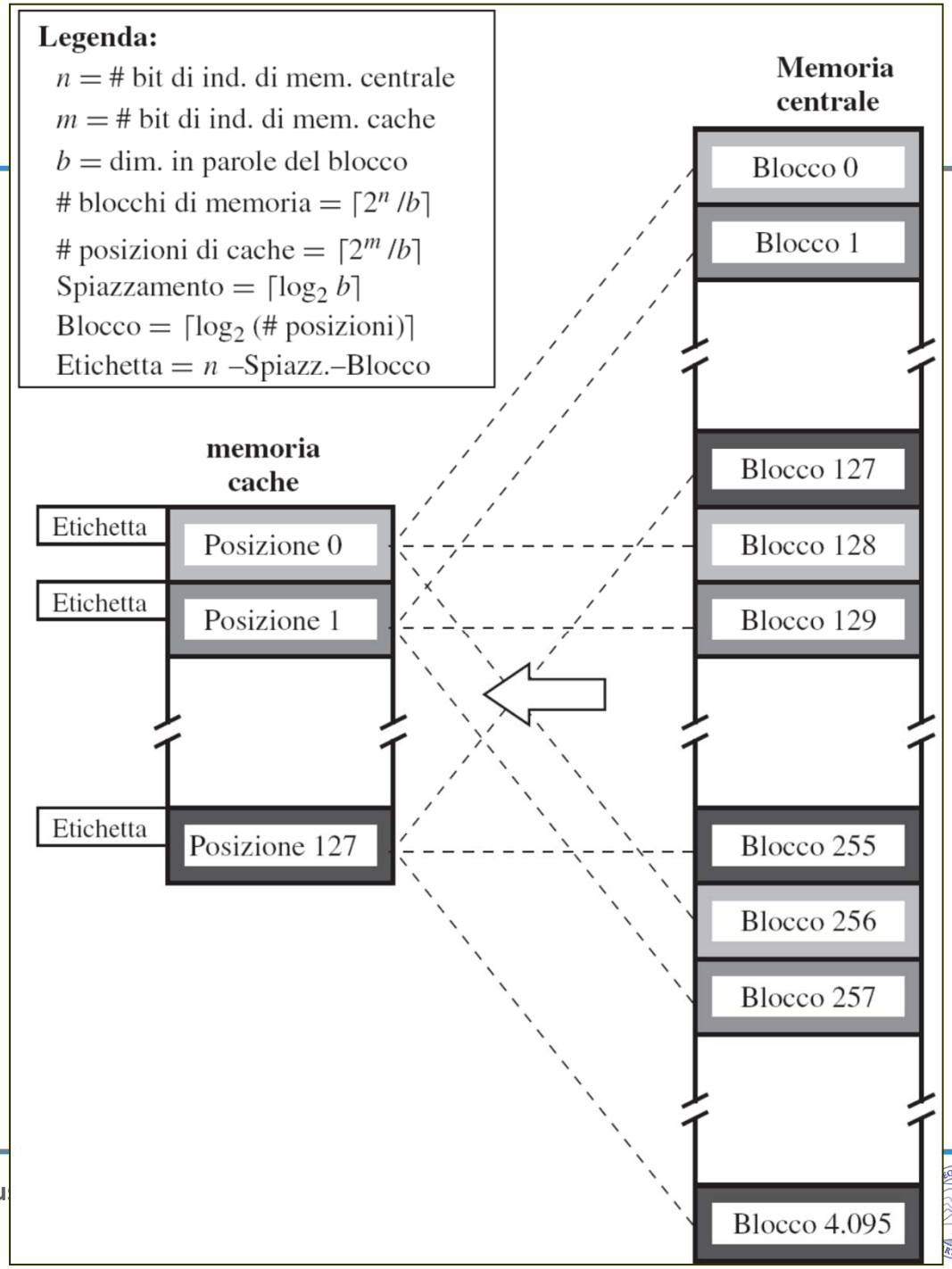
- Si consideri un Sistema con memoria RAM da 64 KB e parole da 1 Byte, dotato di una cache da 2KB con blocchi da 16 parole

- Si disegni lo schema degli indirizzi nel caso delle 3 politiche di mapping

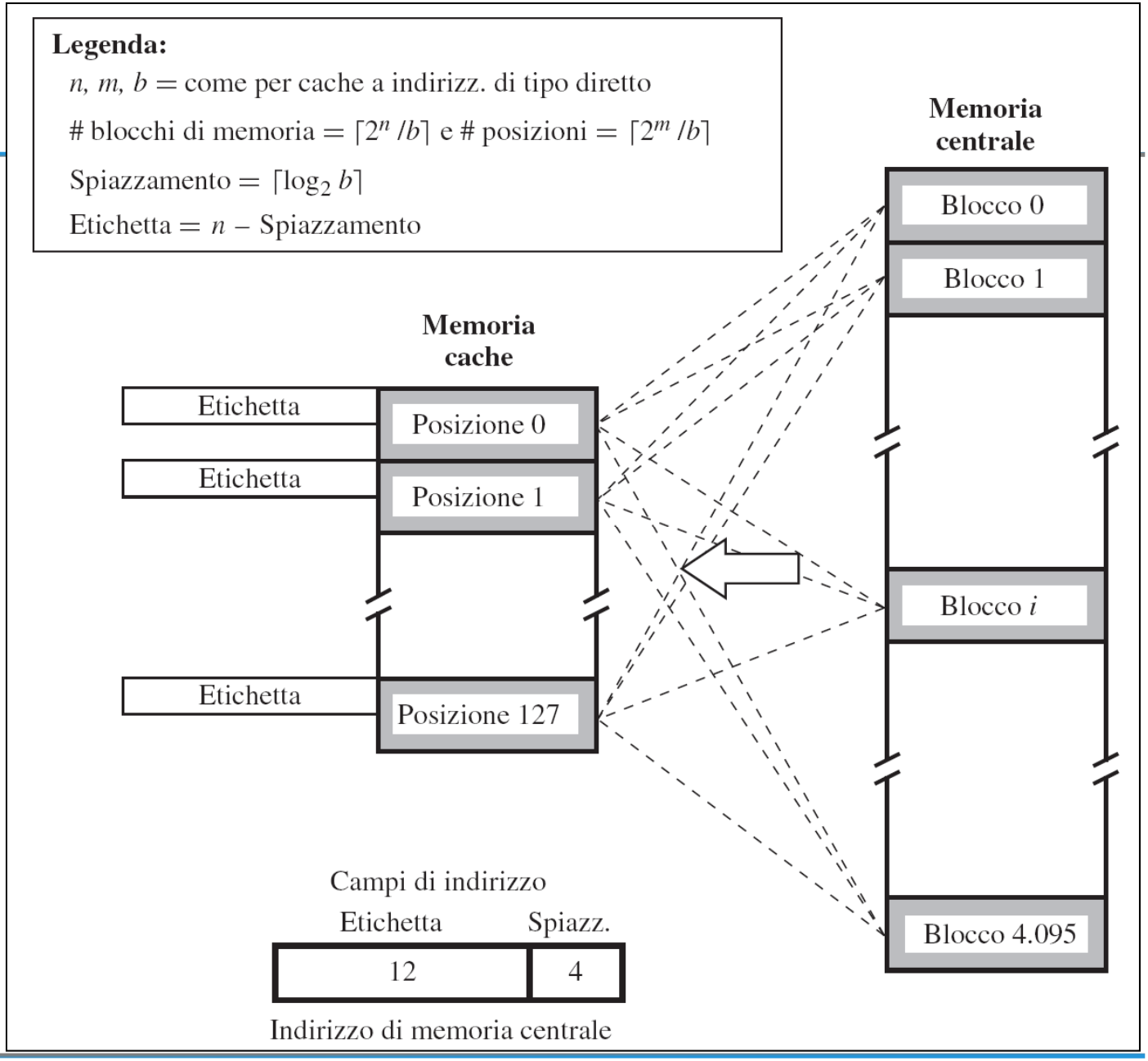
Direct Mapping

etichetta blocco spiazzamento

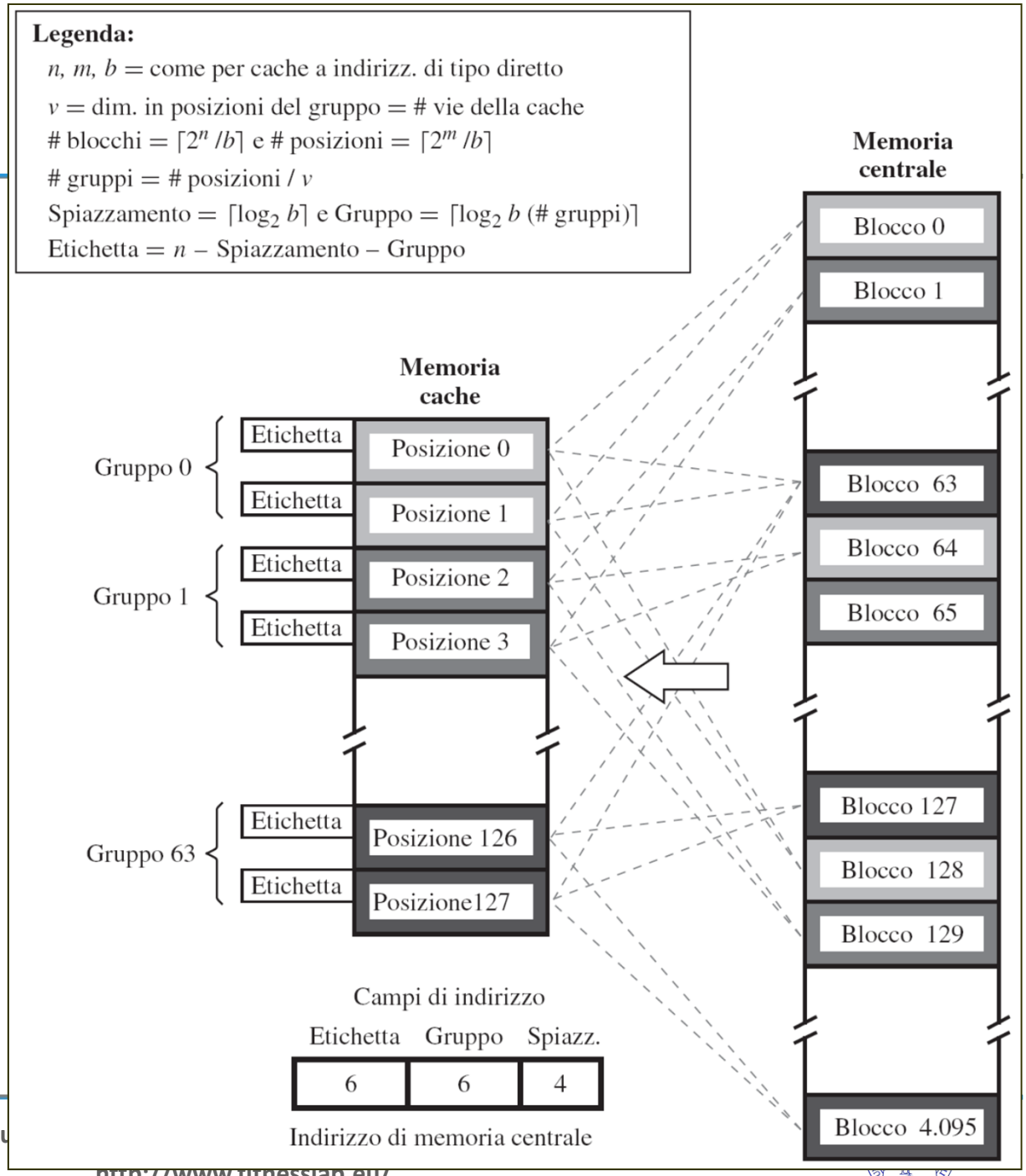
5	7	4



Full Associative



Set Associativa a due vie



Esercizio 2 (Proposto)

- Sia data una cache con:
  - 4K blocchi
  - ogni blocco ha 4 parole
  - indirizzi di 32 bit
- Trovare il numero totale di bit per i tag nel caso di:
  - cache a corrispondenza diretta
  - cache set associativa a 4 vie
  - cache completamente associativa

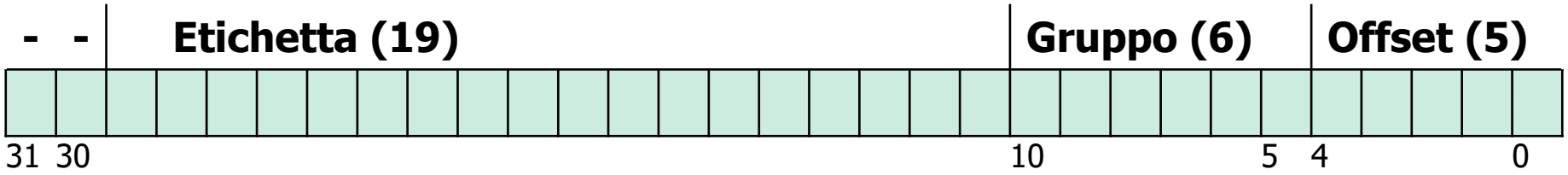
Esercizio 3 (Svolto)

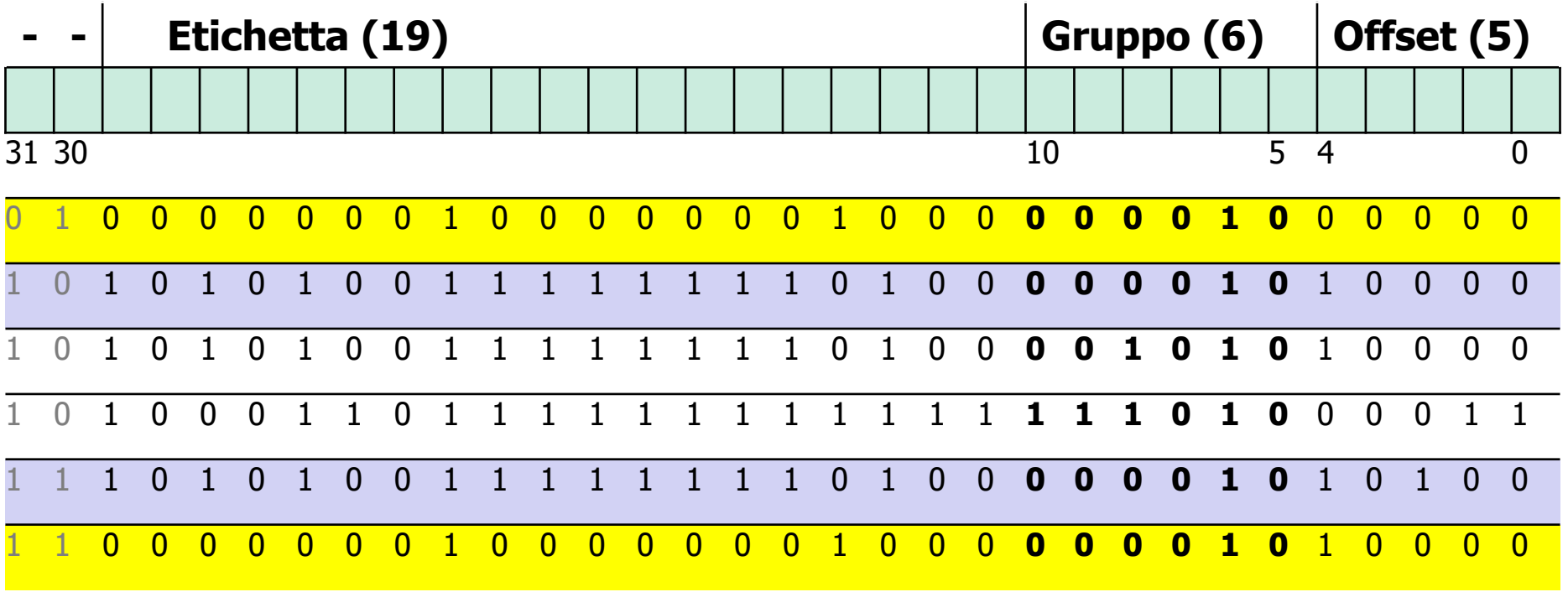
- Si consideri una cache di 16K a 8 vie e blocchi da 8 word, per un Sistema con indirizzamento a byte e word di 4 byte; address bus a 30 bit e memory address a 32 bit.

- In quale Gruppo e con quale etichetta viene memorizzato l'indirizzo 0x40404040?
- Si consideri la seguente sequenza di indirizzi, quanti hit genera supponendo la cache inizialmente vuota?
  - 0x40404040 0xA37FFF43
  - 0xAA7FA050 0xEA7FA054
  - 0xAA7FA150 0xC0404050

Organizzazione Indirizzo

- 1 blocco=8 word di 4 byte = 2^3*2^2=2^5 (5 bit offset)
- 1 Gruppo = 8 vie da 2^5 byte = 2^3*2^5=2^8
- Cache da 16K = 2^4*2^10=2^14
- #gruppi = 2^14/2^8=2^6 gruppi (6 bit di gruppi)




- -
Etichetta (19)
Gruppo (6)
Offset (5)
31 30
10
5
4
0
0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0
1 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0
1 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0
1 0 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1
1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0
1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0

Esercizio 4 (Proposto)

- Si consideri una cache 1-way associative (diretta) e si assuma che l'indirizzo fisico sia di 16 bit, suddiviso in 4 bit di OFFSET, 4 bit di INDEX e 8 bit di TAG. Si consideri inoltre la sequenza di accessi in lettura:
  1. 0x7F5A
  2. 0x3CC7
  3. 0x7F5B
  4. 0x10C2
  5. 0x8F50

a) Quali saranno i blocchi validi alla fine della sequenza di letture?

b) Si verificano conflitti durante la sequenza di lettura? Se sì, quali?

Riferimenti

- Computer Organization and Embedded Systems, 6th Ed., Carl Hamacher, Chap. 8
- Computer Architecture, A Quantitative Approach, John L. Hennessy and David A. Patterson, Chap. 2 and App. B

- https://www.cpuid.com/softwares/cpu-z.html
- http://www.7-cpu.com/cpu/Skylake.html